IA de réunion entièrement locale : ce qui arrive avec Hedy 3.2, et ce que cela coûte en vitesse

Une plongée technique dans l'IA sur l'appareil de Hedy 3.2 : les modèles que nous avons choisis, comment ils tiennent sur Mac, Windows et iPhone, et ce que l'inférence locale coûte en vitesse.

Publié par Julian Pscheid · 22 avril 2026
Lire cet article en ligne: https://www.hedy.ai/fr/post/local-ai-engineering-deep-dive-hedy-3-2/

Un ingénieur logiciel à un bureau en bois étudie attentivement un MacBook orienté à l'écart de la caméra, avec un délicat hologramme cyan et violet qui s'élève du clavier et suggère un calcul IA effectué sur l'appareil

Hedy 3.2 peut faire passer une réunion entière par votre ordinateur portable sans que rien ne quitte l'appareil : audio, transcription, résumés, notes, suggestions, tout en local. Le cloud reste notre option par défaut, et pour la plupart des utilisateurs, c'est toujours le bon choix. Le local est pour les personnes qui préfèrent garder leurs conversations sur leur propre matériel, même quand cela implique d'accepter un coût en vitesse (et sur du matériel très haut de gamme, parfois pas). Pour la première fois, nous pensons que ce compromis est assez honnête pour être livré.

Hedy est un coach de réunion utilisé par environ 30 000 personnes sur Mac, Windows, iOS, Android et le Web. La reconnaissance vocale fonctionne en local sur chaque plateforme depuis le premier jour. La couche d'analyse IA (résumés, notes détaillées, chat pendant la session, suggestions en direct de type « que devrais-je dire ensuite ») était toujours cloud. C'est ce qui change avec la version 3.2.

Cet article explique ce qui a rendu cela possible en 2026, les modèles que nous avons choisis, et ce que nous avons délibérément choisi de ne pas construire. Pour la perspective côté utilisateur sur ce que l'IA locale signifie pour votre workflow de réunion — confidentialité, cas d'usage pour les avocats, les professionnels de santé et les journalistes, et comment l'activer — consultez notre vue d'ensemble de l'IA locale pour les réunions (/fr/post/local-ai-meetings-hedy-3-2/) .

Les deux courbes

Il y avait deux lignes de tendance que nous suivions depuis quelques années.

La première était la qualité des modèles à poids ouverts. Llama 2, mi-2023, était une curiosité pour tout ce qui dépassait une démo de chatbot. Fin 2024, Llama 3 et Qwen 2.5 étaient réellement utiles à petite taille, mais on sentait encore l'écart sur des tâches comme le résumé de réunions multi-tours, où le modèle doit suivre qui a dit quoi sur trente minutes de dialogue. Pendant 2025 et jusqu'au début de 2026, deux familles ont pris de l'avance pour notre cas d'usage : Gemma 4 de Google et Qwen 3.5/3.6 d'Alibaba. Les deux équipes ont consacré un vrai effort à ce qu'on pourrait appeler la densité par paramètre, et à elles deux elles couvrent tout, des modèles sur l'appareil de classe 2B jusqu'à un MoE 35B, avec plusieurs tailles qui font mieux que ce que leur poids suggère en suivi d'instructions et en sortie structurée.

La deuxième courbe était le matériel grand public. L'architecture de mémoire unifiée d'Apple Silicon s'est révélée avoir exactement la bonne forme pour l'inférence de transformers : les poids du modèle vivent à côté du calcul GPU, sans aller-retour PCIe, sans budget VRAM séparé. À la génération M3, un MacBook Air de base avec 16GB pouvait exécuter un modèle quantifié 8-9B à des vitesses utilisables. Windows a suivi un autre chemin. Les GPU discrets avec 8-12GB de VRAM sont courants dans toute machine vendue pour le gaming ou la création de contenu, et les frameworks d'inférence basés sur Vulkan extraient désormais la majeure partie de cette performance du matériel NVIDIA, AMD et, de plus en plus, Intel.

Les deux courbes se sont croisées début 2026. Nous pouvions choisir un modèle qui faisait bien notre travail, et il existait une base installée de matériel grand public capable de l'exécuter. Alors nous l'avons livré.

Ce que nous avons choisi, et pourquoi

Le runtime sous-jacent est llama.cpp (https://github.com/ggml-org/llama.cpp) , intégré comme submodule et compilé par plateforme avec le backend approprié. Nous utilisons des poids quantifiés GGUF issus des builds HuggingFace d'Unsloth (https://huggingface.co/unsloth) , qui sont actuellement la source la plus propre de modèles à poids ouverts bien quantifiés dans cette gamme de tailles.

Le catalogue actif au lancement est le suivant :

Modèle | Quant | Taille du fichier | RAM au chargement | Qualité

Qwen 3.5 2B | Q4_K_M | 1.2 GB | 3 GB | &
Qwen 3.5 4B | Q4_K_M | 2.6 GB | 4.5 GB | &
Gemma 4 E2B | Q4_K_M | 2.9 GB | 5 GB | &
Gemma 4 E4B | Q4_K_M | 4.6 GB | 7 GB | & &
Qwen 3.5 9B | Q4_K_M | 5.3 GB | 8 GB | & &
Qwen 3.5 9B HQ | Q8_0 | 8.9 GB | 12 GB | & &
Qwen 3.6 27B | UD-Q4_K_XL | 17.6 GB | 22 GB | & & &
Gemma 4 31B | Q4_K_M | 17.1 GB | 23 GB | & & &
Qwen 3.6 35B-A3B | UD-Q4_K_M | 20.6 GB | 25 GB | & & &

S'y ajoutent quelques SKUs archivés de versions antérieures, qui restent sélectionnables pour les utilisateurs qui les ont déjà téléchargés, mais disparaissent du sélecteur une fois le fichier local supprimé. Le sélecteur réduit tout à une note de qualité sur trois étoiles : une étoile pour les modèles sous 7B, deux pour 7-12B, trois pour plus de 12B. La quantification (Q4 vs Q8) ne change pas la note ; les étoiles parlent de capacité brute, pas de précision de déploiement.

Le préfixe « E » dans Gemma 4 E2B et E4B correspond à la notation de Google pour les « paramètres effectifs » : E2B représente environ 2.3B paramètres effectifs, avec environ 5.1B en comptant les embeddings, et E4B représente environ 4.5B effectifs, avec environ 8B en comptant les embeddings. Les lignes Qwen utilisent des nombres de paramètres totaux directs. Nous trions le sélecteur selon ce qui compte réellement pour les utilisateurs (RAM au chargement et étoiles de qualité), ce qui évite que les deux conventions de nommage se brouillent l'une l'autre.

Pourquoi ces deux familles. Gemma 4 et Qwen 3.5/3.6 sont toutes deux livrées en plusieurs tailles dans la plage qui nous intéresse, toutes deux sont solides en suivi d'instructions à petite taille (ce qui compte parce que le résumé de réunion consiste surtout à « suivre ce schéma de sortie et ne pas halluciner »), toutes deux ont des licences permissives pour une distribution commerciale, et toutes deux ont des tokenizers bien comportés sur les 51 langues d'interface que Hedy prend actuellement en charge. La variante Qwen 27B utilise la quantification UD-Q4_K_XL (« Unsloth Dynamic ») d'Unsloth, qui préserve davantage de précision des poids dans les couches les plus importantes. C'est une amélioration sensible par rapport à un Q4 naïf à taille de fichier égale.

Ce que vous voyez réellement dans l'app est un sélecteur de modèles qui liste les modèles compatibles avec votre machine, avec la taille disque, la RAM approximative au chargement et la note en étoiles. Si un modèle pouvait techniquement s'exécuter mais devait déverser certaines couches vers le CPU sur votre matériel, l'entrée reçoit le suffixe "+ Slow". Nous préférons être honnêtes dans le sélecteur plutôt que de laisser quelqu'un télécharger 5GB puis se demander pourquoi les résumés prennent quatre minutes.

Apportez votre propre modèle. Le sélecteur comporte aussi une section "Custom models" où vous pouvez pointer Hedy vers n'importe quel GGUF compatible sur disque et l'utiliser à côté du catalogue organisé. Nous ne copions pas le fichier ; il reste là où vous l'avez placé, et Hedy conserve une référence persistante pour que le lien survive entre les lancements. Il n'y a aucune garantie de compatibilité : il faut que ce soit un GGUF que llama.cpp peut charger, que le template de chat soit assez sain pour que nos prompts retournent une sortie structurée utile, et vous êtes responsable de votre budget mémoire. Mais si vous êtes le genre de personne qui lit ce paragraphe et se dit « je pourrais essayer DeepSeek V4 là-dessus », allez-y.

Apple Silicon : le cas facile

Apple Silicon est la plateforme où l'IA sur l'appareil donne déjà l'impression que le futur est arrivé. Trois choses la font fonctionner.

D'abord, le modèle de mémoire unifiée. Le GPU et le CPU partagent le même pool de RAM. Charger en mémoire un modèle quantifié de 8B paramètres signifie que le GPU peut lire ces poids à la bande passante mémoire, sans aucun des coûts de copie vers la VRAM qui sont standards dans une configuration Windows avec GPU discret. Pour l'inférence, où vous faites circuler les poids dans le calcul des milliers de fois par token, c'est un avantage structurel significatif.

Ensuite, Metal. L'API de calcul GPU d'Apple est mature, bien documentée, et le backend Metal de llama.cpp est l'un de ses plus aboutis. Nous compilons avec GGML_METAL=ON et nous n'ajoutons rien d'exotique par-dessus.

Enfin, le plancher des puces M-series. Même un M1 de base dispose de suffisamment de GPU et de bande passante mémoire pour exécuter de petits modèles quantifiés à des vitesses interactives.

En pratique, cela se traduit par les niveaux du catalogue :

• Compact (Macs 8GB) : Gemma 4 E2B, Qwen 3.5 4B, Qwen 3.5 2B. Qualité une étoile. Suffisant pour les résumés courts et les sorties structurées, plus faible sur les longues réunions.

• Standard (Macs 16GB) : Gemma 4 E4B, Qwen 3.5 9B. Qualité deux étoiles. Le point d'équilibre pour la plupart des utilisateurs, y compris les MacBook Air en configuration de base. La qualité se rapproche d'un petit modèle frontier hébergé sur notre tâche.

• Standard, précision supérieure (Macs 16GB+) : Qwen 3.5 9B HQ. Le même modèle que le Qwen 9B du niveau Standard, en Q8 au lieu de Q4. Même nombre de paramètres, plus de précision des poids par couche. Un saut de qualité plus petit que le passage à une classe de paramètres supérieure, mais réel si vous avez la marge.

• Pro (Macs 24-32GB) : Qwen 3.6 27B, Gemma 4 31B. Qualité trois étoiles. Les deux fichiers font environ 17GB sur disque et demandent 22-23GB de RAM au chargement, avec une marge de KV cache pour une longue réunion.

• Max (Macs 32GB+) : Qwen 3.6 35B-A3B. Qualité trois étoiles. Architecture MoE (35B paramètres totaux, ~3B actifs par token), ce qui la rend nettement plus rapide en inférence que son nombre de paramètres ne le suggère. Demande environ 25GB au chargement.

La plus agréable surprise a été le comportement du niveau Standard sur un MacBook Air en configuration de base. Nous étions prêts à dire aux utilisateurs que l'IA locale était réservée aux power users. Sur Apple Silicon, ce n'est pas le cas.

Windows : la VRAM et le problème du débordement

Windows est plus compliqué. La variance matérielle est énorme : un développeur avec une tour gaming récente et une 4080 a plus de puissance d'inférence que n'importe quel Mac que nous expédions, tandis qu'un travailleur de la connaissance sur un ordinateur portable professionnel à graphiques intégrés n'en a pratiquement aucune.

Nous avons pris une position assumée sur le build Windows : Vulkan est requis pour l'IA locale . Le build Windows CMake définit GGML_VULKAN=ON quand le SDK Vulkan est présent au moment de la compilation, et saute entièrement la cible llama.dll s'il ne l'est pas. Il n'y a pas de fallback CPU-only pour l'IA locale sur Windows. CUDA serait plus rapide sur NVIDIA, mais verrouiller l'expérience sur un seul fournisseur de GPU aurait signifié exclure tous les utilisateurs AMD et Intel. Vulkan nous rapproche de CUDA sur NVIDIA, et c'est le seul chemin qui fonctionne avec les fournisseurs de GPU que notre base d'utilisateurs possède réellement.

La forme du problème sur Windows est différente de celle du Mac. Les GPU discrets ont leur propre VRAM, séparée de la RAM système, et le modèle doit tenir dans cette VRAM pour bénéficier d'une accélération GPU complète. Si le modèle est légèrement trop gros, llama.cpp peut le diviser : la plupart des couches vont sur le GPU, le reste tourne sur le CPU. Cela fonctionne, mais chaque token attend désormais la couche la plus lente de la chaîne. Nous avons vu des cas où passer de « tient dans la VRAM » à « deux couches sur le CPU » réduisait le débit d'environ moitié. Dès que vous débordez de plus d'une poignée de couches, autant tourner sur CPU.

Calculer correctement cette répartition compte. Chaque modèle du catalogue a son numLayers enregistré (24 pour Qwen 2B, 32 pour Qwen 4B, et ainsi de suite, récupéré dans le config.json HuggingFace de chaque modèle). Passer une valeur irréaliste à n_gpu_layers de llama.cpp ne vous apporte rien : llama.cpp plafonne l'offload au nombre réel de couches du modèle, et toute prédiction de compatibilité ou tout calcul de débordement contre une valeur gonflée obtient silencieusement la mauvaise réponse. Nous gardons donc ces chiffres honnêtes dans le catalogue.

Notre approche dans le sélecteur de modèle est d'être précis sur le résultat. Nous vérifions la VRAM disponible, calculons l'empreinte du modèle (poids plus KV cache plus buffer de travail), puis :

• Si le modèle tient, nous le listons normalement.

• S'il déborde modestement vers le CPU, nous ajoutons le suffixe "+ Slow", pour que les utilisateurs sachent qu'ils doivent s'attendre à une latence notable.

• S'il débordait de façon catastrophique, nous ne le listons pas du tout sur ce matériel.

Si la machine n'a aucun GPU détecté, ou une mémoire insuffisante même pour le plus petit modèle, la section d'IA locale dans les paramètres affiche une raison de blocage au lieu d'un sélecteur.

Un détail Windows spécifique mérite d'être mentionné : la version de pilote NVIDIA 595.xx avait un problème connu avec le calcul Vulkan sur les cartes RTX 40-series sur certains builds Windows, qui provoquait des crashs FAST_FAIL pendant le chargement du modèle. Nous avons enquêté en pensant avoir livré un bug, puis nous avons compris que la même signature de crash apparaissait dans des applications Vulkan sans rapport sur la même version de pilote. Le correctif était upstream, dans le pilote NVIDIA 596.21. Nous affichons maintenant un message d'erreur plus clair, mais le vrai correctif est « mettez votre pilote à jour ».

Mobile : plus petit niveau, honnête sur l'écart

Nous n'avons activé l'IA locale que sur iPhone 15 Pro et ultérieurs (puce A17 Pro et au-delà), ainsi que sur les iPad M-series. Le raisonnement est simple : tout ce qui est plus ancien n'a ni la mémoire unifiée ni la génération de Neural Engine nécessaires pour exécuter même le plus petit niveau à des vitesses acceptables.

Même sur un 15 Pro, vous exécutez des modèles du niveau Compact : Qwen 3.5 2B, Qwen 3.5 4B, Gemma 4 E2B. Un modèle quantifié de 2-5B paramètres est une autre catégorie de rédacteur qu'un modèle 9B. Bon en sortie structurée, bon pour suivre des instructions claires, correct pour les résumés courts. Il perd le fil dans les longues réunions, surtout celles qui comportent beaucoup d'intervenants ou de jargon technique. Nous sommes transparents à ce sujet dans l'UI mobile : le modèle local est étiqueté comme tel, et nous recommandons Cloud AI comme option par défaut pour l'analyse sérieuse de réunions sur téléphone.

Android et le Web n'ont pas de mode IA locale visible par l'utilisateur dans la version 3.2. La plomberie FFI Android existe dans le codebase (libllama.so est compilé), mais nous n'avons pas encore livré d'UI testée et contrôlée pour cela. La variance matérielle sur Android et les contraintes d'exécution (pas d'équivalent de l'intégration serrée de Metal) rendent difficile la promesse d'une expérience cohérente aujourd'hui. Le Web est hors sujet pour l'instant : les contraintes du navigateur autour du stockage persistant, de l'accès au calcul GPU et des limites mémoire restent des cibles mouvantes. Les deux viendront probablement plus tard. Aucun des deux n'est engagé.

Ce qui reste local, et ce qui ne l'est pas

L'objectif de livrer l'IA locale est que quelqu'un puisse faire passer une réunion par Hedy sans que rien concernant cette réunion ne quitte sa machine. Nous avons pris cet objectif au sérieux, ce qui implique d'être précis sur ce qui circule où.

Avec l'IA locale activée et Cloud Sync désactivé :

• Capture audio : locale, jamais téléversée.

• Transcription : locale, jamais téléversée.

• Résumés, notes détaillées, chat pendant la session, suggestions en direct : modèle local, aucun appel API pour le travail IA.

• Stockage de session : sur l'appareil.

• Rien concernant la réunion ne touche nos serveurs.

Avec l'IA locale activée et Cloud Sync activé :

• Tout ce qui précède, plus :

• Les données de session (transcriptions, résumés, métadonnées) se synchronisent avec notre backend, chiffrées en transit et au repos.

• C'est ce qui permet l'accès entre appareils. Votre téléphone peut afficher les notes d'une réunion que votre ordinateur portable vient de résumer.

• Le travail IA lui-même reste local. Cloud Sync est une couche de synchronisation, pas un chemin d'inférence de fallback.

Toujours, quels que soient les paramètres :

• Les informations de compte (email, état d'abonnement) passent par nos serveurs. Il n'existe pas de façon pour un produit avec compte d'exister sans compte.

• La télémétrie d'usage anonyme et les rapports de crash vont vers notre monitoring. Rien de cela ne contient le contenu des conversations.

• Les vérifications de mise à jour de l'app, les feature flags, les téléchargements de modèles (qui viennent d'un CDN). Des GET HTTPS de contenu quasi public, sans payload par utilisateur.

Ce que nous avons délibérément choisi de ne pas construire, c'est un fallback cloud silencieux. Si vous avez activé l'IA locale et que le pipeline local échoue (le modèle crashe, manque de mémoire en pleine génération, lance une erreur d'inférence), l'erreur remonte à l'appelant. Il n'y a pas de nouvelle tentative discrète auprès de nos serveurs. Quelqu'un qui a activé l'IA locale l'a fait pour une raison. Basculer vers le cloud sans le lui dire trahirait l'hypothèse faite au moment de l'activation.

Ce que cela coûte

Cloud AI reste meilleur pour la plupart des utilisateurs. Nous devons le dire clairement, parce que le public technique de cet article verra à travers n'importe quel marketing sur ce point.

Le cloud est plus rapide pour la plupart des utilisateurs, souvent d'un ordre de grandeur. Un résumé terminé en quelques secondes dans notre pipeline cloud peut prendre 30 secondes en local sur un modèle de niveau Standard, et plusieurs minutes si l'utilisateur a choisi un modèle de niveau Pro sur une machine limite. Les suggestions en direct, qui se déclenchent pendant que la réunion est en cours, sont nettement plus réactives en mode cloud.

Il y a une exception honnête qui mérite d'être signalée. Sur un MacBook Pro très haut de gamme (par exemple un M5 Max avec 128GB de mémoire unifiée) exécutant un modèle local solide, la comparaison peut s'inverser. L'inférence cloud n'est pas elle-même exempte de problèmes de latence : les modèles hébergés partagent la capacité GPU entre de nombreux utilisateurs, et le time-to-first-token peut exploser quand la charge est forte ou qu'un modèle populaire est en file d'attente. Un modèle local déjà chargé en mémoire n'a aucune de ces variations ; il tourne à la vitesse que votre matériel lui donne, de façon déterministe, à chaque fois. Pour les utilisateurs dotés d'un matériel capable qui exécutent de courtes requêtes pendant que le cloud est chargé, le local peut réellement gagner en temps réel. Nous ne voulons pas surpromettre (la plupart des utilisateurs sur la plupart des matériels verront toujours le cloud comme plus rapide en moyenne), mais cela vaut la peine de le dire clairement : la comparaison de vitesse n'est pas unidirectionnelle.

Le cloud fonctionne sur toutes les plateformes. Le local, non. Si vous êtes sur Android, sur le Web, ou sur un Mac ou un iPhone plus ancien, le mode local n'est pas encore une option pour vous.

La raison pour laquelle la plupart des utilisateurs choisiront le local est la confidentialité, pas la performance. Nous avons essayé de rendre le local assez bon pour que le compromis de confidentialité soit réel, pas un geste, mais sur la plupart des matériels, cela reste un compromis. Choisir le local parce que vous voulez vos réunions sur votre machine est rationnel. Le choisir pour la vitesse n'a de sens que si votre matériel peut suivre.

Une note spécifique : Automatic Suggestions, la fonctionnalité qui exécute le LLM en continu pendant une réunion pour anticiper quoi dire ensuite, est lourde. C'est la charge de travail IA la plus coûteuse de Hedy. Dans le cloud, vous ne le remarquez pas. En local, surtout sur une petite machine, elle peut saturer la boucle d'inférence et ralentir le reste de votre ordinateur. L'app affiche un dialogue d'orientation unique quand vous activez Local AI Processing : "Automatic Suggestions run the AI continuously during a session. With Local AI Processing, this can keep your CPU and GPU busy and slow down the rest of your computer. You can turn them back on any time in Settings." Nous avons envisagé de désactiver Automatic Suggestions de force en mode local et avons décidé de ne pas le faire : quelqu'un avec une machine haut de gamme peut l'exécuter, et nous préférons ne pas infantiliser.

Décisions d'architecture qui valent la peine d'être discutées

Quelques éléments que nous avons faits délibérément et que le lecteur technique pourrait trouver intéressants.

Pas de fallback silencieux. Déjà mentionné. Le coût côté utilisateur est parfois une erreur confuse. Le bénéfice est une fonctionnalité qui veut dire ce qu'elle dit.

Configuration par appareil. Les paramètres d'IA locale (quel modèle est sélectionné, quelles fonctionnalités utilisent le local) ne se synchronisent pas entre appareils. Votre Mac peut avoir un modèle 9B installé ; votre téléphone a un modèle 2B ; ils sont configurés séparément. La synchronisation serait incorrecte ici : le bon modèle dépend du matériel sur lequel vous êtes, pas de ce que l'utilisateur préfère dans l'abstrait.

Les modèles archivés restent sélectionnables jusqu'à suppression. Quand un modèle du catalogue est remplacé (Qwen 3.5 27B !’ Qwen 3.6 27B, par exemple), nous marquons l'ancienne entrée comme archivée au lieu de la supprimer. Les utilisateurs qui l'ont déjà téléchargée peuvent continuer à l'utiliser ; une fois qu'ils suppriment le fichier local, l'entrée disparaît entièrement de l'UI. Nous ne sommes pas là pour briquer un téléchargement de 17GB parce que le catalogue a avancé.

La quantification comme choix organisé, pas comme bouton. Unsloth livre chaque modèle dans une douzaine de variantes de quantification : Q2 through Q8, avec des saveurs K, K_M, K_L, XL et des versions dynamiques. Les exposer toutes serait un piège. La plupart des utilisateurs ne peuvent pas distinguer Q4_K_M de UD-Q4_K_XL, et la chute de qualité dans le bas de gamme est raide. Nous choisissons des quantifications spécifiques par modèle selon le compromis taille-qualité pour ce modèle précis. Q4_K_M pour la plupart. L'UD-Q4_K_XL dynamique d'Unsloth pour le 27B, où la quantification consciente des couches récupère une vraie qualité à taille de fichier égale. Une variante Q8_0 "HQ" du 9B pour les utilisateurs qui ont de la marge en RAM et veulent l'expérience la plus proche de fp16 disponible à cette classe de paramètres. Le libellé "HQ" est délibéré ; afficher "Q8_0" comme chaîne d'UI signifierait enseigner à chaque utilisateur ce qu'est la quantification GGUF, et ce n'est pas un coût que nous voulons imposer.

Compatibilité et vitesse sont calculées, pas devinées. Chaque entrée du catalogue passe par un algorithme de scoring contre le matériel détecté avant d'apparaître dans le sélecteur. Le scorer estime la mémoire requise (poids + KV cache + buffer de travail à la quantification de catalogue du modèle) par rapport à ce qui est réellement disponible, et estime les tokens par seconde en utilisant la bande passante mémoire du GPU quand elle est connue (avec des pénalités de mode pour l'offload GPU partiel et l'exécution CPU-only). À partir de là, il produit une classification de compatibilité (Great fit / Tight fit / Won't fit) et un suffixe "+ Slow" quand l'exécution impliquera un offload CPU. Sur Windows, c'est le cas évident : les couches du modèle débordent de la VRAM vers la RAM système. Sur macOS, le même suffixe se déclenche quand le modèle ne tient qu'en poussant l'OS à évincer d'autres apps pour câbler la mémoire des poids, ce qui charge techniquement, mais vous le sentirez. Un badge "Recommended" va au modèle au score le plus élevé qui arrive à Great fit (pas Tight) ; nous ne recommandons délibérément pas les modèles Tight-fit même quand ils auraient un meilleur score en qualité brute, parce qu'orienter quelqu'un vers un modèle qui tourne nettement plus lentement qu'il ne le devrait est pire que l'orienter vers le niveau inférieur qui tourne proprement.

Une annulation qui fonctionne vraiment. La génération llama.cpp tourne dans un isolate worker pour que le thread UI reste réactif, avec le streaming des tokens vers l'isolate principal via un NativeCallable.listener . L'annulation passe par llama_set_abort_callback , qui vérifie un flag atomique, validé contre la source upstream plutôt que deviné, afin qu'un utilisateur qui ferme une session au milieu d'une longue génération arrête réellement le travail, au lieu de le laisser se terminer en arrière-plan et gaspiller de la batterie.

Ce que nous ne savons pas encore

La liste honnête :

• Nous n'avons pas encore d'excellents chiffres de latence en conditions réelles pour une large gamme de matériels. Nous avons testé en profondeur sur les machines que nous possédons, interrogé des bêta-testeurs et lu les rapports de crash, mais la longue traîne du « laptop Windows avec graphiques intégrés de 2018 » est difficile à caractériser. Les premières données de production nous diront où se trouvent réellement les aspérités. Dans nos propres tests, nous avons vu l'inférence locale prendre de cinq secondes à cinq minutes, selon la longueur de la réunion, le contexte et le modèle sélectionné.

• Nous ne connaissons pas encore la bonne cadence pour livrer de nouvelles versions de modèles. Le churn du catalogue est déjà réel (Qwen 3.5 !’ 3.6 en un cycle). Jusqu'ici, nous l'avons géré en archivant plutôt qu'en supprimant, mais si l'usage disque cumulé devient incontrôlable, nous devrons peut-être adopter une autre politique.

• La consommation d'énergie sur les ordinateurs portables va surprendre certains utilisateurs avec les charges de travail IA continues. Une réunion avec Automatic Suggestions exécuté en local est nettement

plus exigeante que la même réunion avec Cloud AI. Nous n'avons pas encore construit de throttling sensible à la batterie, et nous devrions probablement le faire. Le cas où quelqu'un veut simplement un résumé à la fin de la réunion est différent : c'est une génération unique, et sur toute machine capable d'exécuter confortablement un modèle de niveau Pro ou Max, elle reste largement dans les marges habituelles de batterie et de thermique.

Où cela nous mène

L'histoire intéressante n'est pas Hedy 3.2 en soi. C'est que les courbes continuent de bouger.

Les modèles à poids ouverts continuent de s'améliorer à petite taille. La classe 2B d'aujourd'hui se situe à peu près là où était la classe 7B il y a dix-huit mois. Quels que soient les douze prochains mois de Gemma, Qwen et des autres familles sérieuses à poids ouverts, le plancher de « ce qui tourne sur un ordinateur portable normal » continue de monter.

Le matériel grand public continue de gagner en capacité. Les générations de puces d'Apple ont discrètement intégré l'inférence dans la roadmap du silicium. Windows commence à livrer des NPU qui finiront par compter pour l'inférence, même si la première génération actuelle relève surtout du marketing.

La combinaison signifie qu'un changement discret est en cours. L'IA signifiait auparavant « une poignée d'entreprises exploitent de grands modèles pour vous, et vous leur envoyez vos données ». Cela devient « une poignée d'entreprises entraînent de grands modèles, mais vous pouvez les exécuter sur votre propre appareil avec vos propres données, de bout en bout ». La version hébergée reste meilleure aujourd'hui, et le restera probablement pendant un moment. Mais l'écart se réduit, et pour un sous-ensemble significatif d'utilisateurs, « assez bon et local » bat « le meilleur et distant ».

Hedy 3.2 est notre premier pari concret sur ce changement. L'architecture est construite pour nous permettre de tirer davantage ce levier à mesure que les modèles et le matériel le permettent. Il y en aura d'autres.

Hedy AI · Coaching IA en direct pour les conversations importantes

Essayez Hedy gratuitement: https://www.hedy.ai/fr/downloads/

https://www.hedy.ai/fr/post/local-ai-engineering-deep-dive-hedy-3-2/